Arguably Cavey was the first true game the Oliver Twins wrote. Sure they had written a few published titles before this, but this was their first 'real game' so I thought I'd ask how and why they developed it, and what they learnt from it.

Chris Wilkins: Thank you for taking the time to talk to me again. Is it fair to say Cavey was your first game?

Philip Oliver: Well it was certainly the first arcade game we wrote. Our previous games were all written in BASIC and were nowhere near as good as this

Andrew Oliver: That makes it sound like it was groundbreaking. It was for us, but not in terms of the best games at the time. In fact, I remember we took it into school to the lunchtime computer club we were running. They had a BBC Micro, so we loaded it up to show people and we were very disappointed when the kids said they'd rather play Chuckie Egg, Snapper, Planetoids, Rocket Raid, Mr EE, or the recently released and technically stunning Revs.

CW: So when was this?

PO: This would have been September 1984. We were new to the sixth form which gave us the privilege to run the computer club and we had written Cavey during the summer holidays after finishing our O Levels.

Above: The rather fetching landing screen of the game.

AO: We were hoping to inspire some of them to learn how to write games and perhaps get some feedback on our games, but they were there to play games, get out of the rain, and as for feedback, well they would have only been impressed if we had written Elite.

PO: Actually, I'm not even sure they would have been impressed by that. They wouldn't have understood how technically amazing that game was and were very critical of everything.

CW: So what led you to write the game and what challenges did you face?

AO: By 1984 we'd mastered BASIC on our BBC Micro — the model B version with 32KB RAM, but it was fairly slow. We wanted to write 'arcade games', but this meant writing machine code, well assembler, and this was a black art. Obviously it was well before the internet so you needed to find a book on the subject or know someone that had already learnt it and was prepared to share the knowledge.

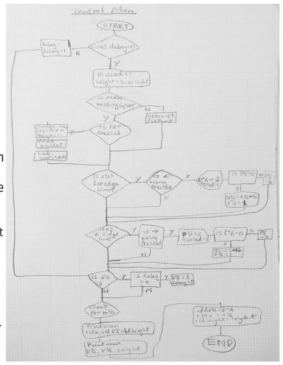
PO: Luckily we were studying Computer Studies O Level and the lecturer, Mr McCann, knew 6502. It wasn't on the curriculum to teach, but we asked him if he could advise us on converting a paint fill routine from BASIC to Assembler for our Easy Art graphic application. He took a printed listing of the BASIC code and converted it into 6502 assembler over multiple pages of lined A4.

AO: It was VERY scruffy, full of errors, but we attempted to type it in and get it running. We didn't understand the instructions, but through trying to get it to compile we managed to work it all out and finally got it running and it was VERY fast. When we talked to Mr McCann about this he talked us into buying the BBC B Advanced User Guide which explained 6502 Assembler in greater depth — although not how to use it for games or sprites, that was down to us to work out.

PO: It was a very expensive purchase at £20 maybe more — that's the equivalent of £60 in today's money, so very expensive for school kids. But it was an excellent book and something we needed.

CW: So why Cavey?

PO: We didn't want to try anything too ambitious but did want to create a full arcade-style game. We thought something similar to Galaga, or Zalaga as it was known on the BBC, as it was about the right amount of challenge. But since that existed, we thought we needed to present it in a completely different way.

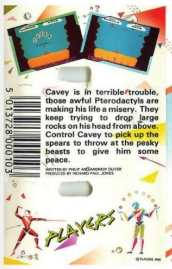


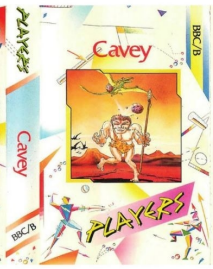
Above: A design page of the game — old school flowchart!

AO: Since that game was about space ships and aliens which made it very sci-fi, we thought about doing the same kind of gameplay only making it set in prehistoric times. The player controlled a trapped caveman who is pelted by rocks dropped by pterodactlys. Luckily our caveman, Cavey, has some spears to attack back — only he has to be careful not to get impaled on his own returning weapons.

CW: So how did you go about making it?

PO: We planned it all





LOADING
1) If your BBC is fitted with disc or Econet then type:
*TAPE (RETURN)
2) CHAIN"" (RETURN)
The program will now automatically load and run.

THE GAME
You control a caveman called Cavey. He has been trapped in the Valley of the Doomed and needs your help to fight off the swarming Pterodactyls using only five spears. The Pterodactyls cannot fly low enough to catch Cavey but carry rocks from the surrounding cliffs to drop on him, in order to knock him off his log bridge. Remember that Cavey must pick up spears before he can throw them.

THE CONTROLS
Cavey may be guided with a joystick or from the keyboard.
Z or Joystick left - Moves Cavey Left
X or Joystick right - Moves Cavey Right
RETURN or Fire Button - Throws spear
Additional controls:
S - Sound on
Q - Sound off
COPY - Pause on
DELETE - Pause off

out on graph paper starting with an overall screenshot of what we wanted the game to look like. This helped us understand the screen space and set the size of everything. We were using Mode 2 — which gave us a resolution of 160 pixels wide and 256 deep with 16 colours per pixel. This meant wide pixels.

AO: The BBC has 32KB of RAM, but the screen resolution uses 20KB, so sadly most of this was eaten up with that. In addition, the operating system reserved almost another 2KB, leaving only 10KB for the game.

PO: For the background, the clouds, cliffs and logs we drew just the left-hand side in Easy Art and then wrote some code to cut this out and compress it a little. Then in the game, we wrote code that would take this raw data and print it back on the left side and then reverse print it on the right side.

The Cavey title was also drawn inside Easy Art. We reserved the top eight colours for the rainbow effect. So when the title was displayed, colours 8-15 were altered each frame to create the rainbow effect. Something we'd seen done in the arcade game Mr Do, although we've no idea if this is how they did it. It meant we were restricted down to eight colours for everything else, but that was enough.

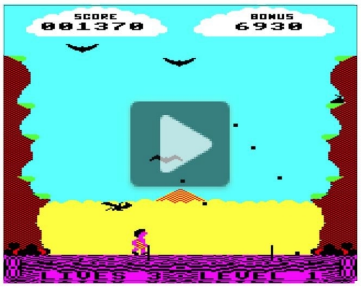
We started drawing the sprites on to graph paper — these consisted of Cavey running and standing animations and the pterodactlys — sitting, taking off, flying and being speared and falling. These had to be converted to data by hand, a pretty long-winded maths job.

CW: I noticed the pterodactlys flicker a little whilst the spears and rocks don't, why is that?

AO: The sprites were displayed and erased using the XOR technique, which means they invert the memory so printing them back in the same place restores the background effectively erasing them. We had to plan the colour palette so whatever they moved over when inverted it would also be black. For this reason, the title with its rainbow writing can't be onscreen at the same time as any of the sprites — as they would have rainbow pixels too.

The spears and rocks were 'hard-coded', they were black pixels moving up and down the screen. Let me explain, if you plot (via XOR) a black pixel and the next frame prints another pixel higher, then repeat each frame you get a vertical line. If you start another pixel plotting say 10 frames later on the same path it acts to erase the line giving the illusion that a black line is travelling up the screen. It's very fast and doesn't flicker. Obviously we made this 'bounce' on the top of the screen and also used the same technique for the falling rocks.

Below: The main playing area with those pesky pterodactlys dropping rocks.

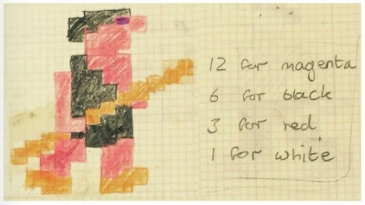


CW: What about the rest of the code?

PO: We used the built-in font and BBC ROM code for printing the text and digits.

AO: We drew out flowcharts identifying the main routines required and the logic required for each.

Because the Assembler source code had to be in the memory the same time as the compiled code, we were already down to 10KB so this gave us memory problems — we therefore had to break the assembler source into multiple chunks, compile the code and



Above: Cavey in all his pixellated graph paper form.

save off chunks of code. Then later we had to write some code to load in all the chunks and ensure they slotted together correctly and were able to address data and jump to pieces of code in other blocks of data. All pretty tricky and it did lead to a chunk of memory being allocated slightly wrong wasting 2KB of RAM — thankfully it wasn't required.

CW: I like the way the pterodactlys fly from their perches and occasionally sweep back and land on the cliff faces. Why and how did you do that?

PO: We wanted to introduce the pterodactlys as living creatures, having them fly from their perch at the start we felt added a lot of personality to them. Getting them to take rests occasionally was a challenge, but it does just about work.

AO: As they leave a perch they read a string of data that defines the path and the sprites to use. When they come off the end of the data they use simple random movement trapped inside a bounding box. To land back on perches they see if they are on the exact coordinates of a path that leads back to a perch — if they are, they then start down that flight path into to a safe landing.

CW: So once finished how did you go about getting it published.

PO: We had a relationship with the leading BBC Publisher at the time, Acornsoft and so sent it to them. They liked it but asked us to speed up the code to ensure it was compatible with the new Electron computer they were bringing out. This was a much cheaper computer that would run the game but at half the speed.

AO: Sadly because we'd had to break the assembler into multiple chunks we'd accidentally lost or written over several bits of code and were unable to recompile the game. Instead, we had to hack into the code and identify a place where we thought we could make some speed improvements. We identified the sprite routine as the area of greatest potential and we were able to change the code to use a lookup table to find the screen coordinate rather than a maths calculation. This turned out to be much faster and we sent a new master to Acornsoft.

PO: They accepted the game for publishing but felt it should be published in the new year (1985). We waited patiently, occasionally reminding them. By September 1986 we wrote to let them know they should publish it or we'll seek other publishers. Sadly they chose not to and we sent it to many other publishers, but by this time the bottom had dropped out of the BBC games market and no one was interested. We had moved to making games for the Amstrad CPC as a result. Another year later our publisher Interceptor decided to set up a budget label and we found an opportunity to present them Cavey. Finally, 2 years after being written it was released in September 1987.

CW: Were you happy with the end result, and with hindsight would you have done anything differently?

AO: Playing it again recently, we shouldn't have made the rocks hit the bottom log and delay before falling off. It was unnecessary and doesn't help the gameplay.

PO: But it is good fun and a real challenge.

CW: Well thanks a lot for your time again, any last words?...

PO: Yes — now you've read about how we made the game — go and play Cavey yourself - in a browser at https://www.olivertwins.com/cavey

AO: We challenge you to beat the third level or more than 20,000.